Práctica 2 – Android: acceso a servicio remoto

Introducción

El objetivo de esta práctica es aprender a comunicarse con una API remota desde una app de Android, y mostrar en la app los resultados obtenidos de dicha API de forma eficiente empleando **RecyclerView**. Para esta práctica es obligatorio conectarse a la API utilizando AsyncTaskLoader como se describe a continuación. **No se permite el uso de librerías externas** para conectarse al servicio.

La app desarrollada mostrará una interfaz sencilla para realizar búsquedas sobre la **API de Google Books**. La interfaz ha de permitir buscar libros y revistas: por autor(es) y/o título si se trata de un libro, y por título exclusivamente si se trata de una revista (en el caso de la revista no se considera autor). De esta manera, la app ha de permitir buscar únicamente libros, únicamente revistas o ambos. Los resultados se mostrarán en una lista utilizando el componente *RecyclerView*, mostrando para cada resultado, como mínimo, el título, el nombre de los autores (si se tratara de un libro, ya que si se trata de una revista no hay que considerar autores y, por tanto, solamente se mostrará el título de la revista), y una URL que permitiría acceder a información más detallada.

Instrucciones

Crear un nuevo proyecto Android usando la plantilla "Empty Views Activity", el nombre de proyecto **GoogleBooksClient**, localizado en el paquete **es.ucm.fdi.pad.googlebooksclient**. Seleccionar Java como lenguaje de programación, y el nivel de API 24 ("Nougat", Android 7.0) para el mínimo SDK.

- Ejecutar en el emulador la aplicación creada por la plantilla y confirmar que se lanza correctamente. Si es necesario crear un nuevo dispositivo virtual.
- Modificar el *layout* de *MainActivity* para crear una interfaz que permita realizar la búsqueda de libros y revistas en base a su título y/o autor (estarán en campos de texto separados), tal y como se ha especificado anteriormente. Utilizar para ello *ConstraintLayout* y *RadioGroup*.
- En un navegador examinar la API de Google Books, en particular la operación Volumen:lista (más detalle en: cómo realizar una búsqueda) que permite buscar libros y revistas en la API de Google Books a través de una consulta de texto. Se deberán obtener unas credenciales para el acceso a la API de Google Books a través de Google Cloud Console.
- **Detalles de implementación (para el ejemplo se utiliza una lista de strings, marcado en amarillo, pero para la práctica se sustituirá por la lista de objetos BookInfo como de describe más adelante):**
  - Crear una clase `BookLoader` que herede de `AsyncTaskLoader<List<String>>` (`import androidx.loader.content.AsyncTaskLoader;`) y devuelva en su método `loadInBackground` el resultado de invocar a la operación `Volumen:lista` de la API de GoogleBooks, para unos `String queryString`, `String printType` dados.
  - En el método `onStartLoading` de `BookLoader` llamar al método `forceLoad()` para forzar siempre la carga del resultado, y no utilizar resultados de búsquedas anteriores.

- Implementar un método `getBookInfoJson(String queryString, String printType)` en la clase `BookLoader` que contenga el código para hacer la petición al servicio usando `HttpURLConnection` y que devuelva el resultado obtenido.
- Para utilizar `BookLoader` desde `MainActivity`:
  - Crear una clase `BookLoaderCallbacks` que implemente la interfaz `LoaderManager.LoaderCallbacks<List<String>>`.
  - El método `onCreateLoader` de `BookLoaderCallbacks` debe devolver una nueva instancia de `BookLoader`.
  - Añadir una variable `private BookLoaderCallbacks bookLoaderCallbacks = new BookLoaderCallbacks()` a `MainActivity`.
  - En el método `onCreate`, inicializar el loader. Se puede implementar con el siguiente código:

```
LoaderManager loaderManager = LoaderManager.getInstance(this);
if(loaderManager.getLoader(BOOK_LOADER_ID) != null){
    loaderManager.initLoader(BOOK_LOADER_ID,null,
                                    bookLoaderCallbacks);
}
```

- Añadir un método `public void searchBooks (View view)` a `MainActivity` y registrarlo como el evento a realizar cuando se pulse sobre el botón de búsqueda. Además, cuando se muestre la lista con los resultados obtenidos se ha de mostrar el texto "Resultados" por encima de la lista. En el método `searchBooks` lanzar el loader con un código similar al que se indica a continuación:

```
Bundle queryBundle = new Bundle();
queryBundle.putString(BookLoaderCallbacks.EXTRA_QUERY, queryString);
queryBundle.putString(BookLoaderCallbacks.EXTRA_PRINT_TYPE, printType);
LoaderManager.getInstance(this).restartLoader(BOOK_LOADER_ID,
                                queryBundle, bookLoaderCallbacks);
```

  - En el código anterior, `queryString` es la concatenación de los nombres de los autores (cuando se trate de un libro y no de una revista) y del título del libro/revista, que se indicaron en las cajas de texto correspondientes de la interfaz de usuario.
  - Para `printType`, se puede utilizar el método `getCheckedRadioButtonId()` de `RadioGroup` para determinar qué `RadioButton` está activo.
- **Sugerencias:**
  - Leer Making an HTTP connection para obtener un ejemplo de cómo realizar una petición a un servicio.
  - Mostrar el string devuelto por el servicio en Logcat para comprobar que la app contacta con el servicio correctamente. Utilizar el parámetro `maxResults` de la petición para limitar el número de resultados devueltos, y que sea más fácil de entender el JSON devuelto.
- Definir una clase `BookInfo` para guardar la información sobre los resultados de la búsqueda, con los siguientes campos como mínimo: `String title; String authors; URL infoLink`. Convertir el string en formato JSON devuelto por la API a una lista de objetos `BookInfo` en un método `static List<BookInfo> fromJsonResponse(String s)` de `BookInfo`. Modificar la clase `BookLoader` para que ahora herede de `AsyncTaskLoader<List<BookInfo>>`, y el método `loadInBackground` devuelva `List<BookInfo>`; y modificar la clase `BookLoaderCallbacks` para que ahora implemente la interfaz `LoaderManager.LoaderCallbacks<List<BookInfo>>`.

- Crear una clase `BooksResultListAdapter` que herede de `RecyclerView.Adapter<BooksResultListAdapter.ViewHolder>` y que sirva de *adapter* de la *RecyclerView*, siguiendo lo visto en clase.
- **Sugerencias:**
  - Añadir una variable `private ArrayList<BookInfo> mBooksData` a `BooksResultListAdapter` que contenga los datos a mostrar. Añadir un método `public void setBooksData(List<BookInfo> data)` para modificar esta variable.
  - Añadir un método `void updateBooksResultList(List<BookInfo> books)` a `MainActivity` para actualizar los datos de la RecyclerView, que llame a los métodos `setBooksData` y `notifyDataSetChanged()` de `BooksResultListAdapter`.
  - Para comprobar que todo funciona bien, llamar al método `updateBooksResultList` con una lista cualquiera en el método `onCreate`, para ver que se visualiza la lista correctamente en la interfaz.
- En `BookLoaderCallbacks` llamar al método `updateBooksResultList` con la lista que se ha pasado cargada.

Partes opcionales

- Utilizar `CardView`, para cada una de las entradas de la `RecyclerView`.
- Añadir un *event handler* para cuando se pulsen las entradas de la `RecyclerView`. De tal forma que utilizando un `Intent` implícito se abrirá la URL de detalle de la entrada en el navegador.
- Mientras se realiza la búsqueda, se ha de mostrar el texto **"Cargando..."** en vez del texto **"Resultados"**, para dar una indicación al usuario de que la aplicación está trabajando para resolver la búsqueda.
- Si una búsqueda devuelve cero resultados, se mostrará el texto **"No se han encontrado resultados"** en vez de **"Resultados"**.
- Realizar una variante con el idioma inglés.